



## Web

Results 1 - 10 of about 2,300,000 for cache class dependency . (0.77 seconds)

### ASP.NET Database Cache Dependency

One of the coolest features of the Cache class is the ability to exert fine-grained control over its behavior with various types of dependencies. ...

[www.eggheadcafe.com/articles/20030716.asp](http://www.eggheadcafe.com/articles/20030716.asp) - 34k - Cached - Similar pages

### ASP.NET Caching Basics

HttpCachePolicy class. The following example shows the code equivalent to the page ...  
NET 1.1, caching based on database dependencies is not built in. ...

[www.eggheadcafe.com/articles/20060407.asp](http://www.eggheadcafe.com/articles/20060407.asp) - 52k - Cached - Similar pages

### Cutting Edge: Implement Custom Cache Dependencies in ASP.NET 1.x ...

So what's a custom cache dependency? It is primarily a class that listens to a data source for changes. When a change is detected, the class bubbles that ...  
[msdn.microsoft.com/msdnmag/issues/04/07/CuttingEdge/](http://msdn.microsoft.com/msdnmag/issues/04/07/CuttingEdge/) - 53k - Cached - Similar pages

### Depend Task

<depend srcdir="\${java.dir}" destdir="\${build.classes}" cache="depcache" closure="yes"/>. removes any classes in the \${build.classes} directory that depend ...  
[ant.apache.org/manual/OptionalTasks/depend.html](http://ant.apache.org/manual/OptionalTasks/depend.html) - 9k - Cached - Similar pages

### Caching in ASP.NET

Using the Cache class in your code-behind class or in the code for the aspx page. Response. ... This is called file dependency or key dependency. Cache. ...  
[www.c-sharpcorner.com/asp/Articles/CachingInASPDPL.asp](http://www.c-sharpcorner.com/asp/Articles/CachingInASPDPL.asp) - 32k - Cached - Similar pages

### Marco Bellinaso's Blog - An excerpt from Chapter 3: Caching Data ...

To create a dependency to a Customers table, you create an instance of the System.Web.Caching.SqlCacheDependency class, whose constructor takes the caching ...  
[www.dotnet2themax.com/blogs/mbellinaso/PermaLink,guid,509e5473-8583-4f32-93d5-92cb8cd39842.aspx](http://www.dotnet2themax.com/blogs/mbellinaso/PermaLink,guid,509e5473-8583-4f32-93d5-92cb8cd39842.aspx) - 93k - Cached - Similar pages

### KinesisSoftware.com :: View topic - Class Caching Technote

When the compiler compiles a class, it will add the class to the class cache with the set of classes that the class depends on. These dependencies are ...  
[www.kinesissoftware.com/FusionForum/viewtopic.php?p=343&sid=19](http://www.kinesissoftware.com/FusionForum/viewtopic.php?p=343&sid=19) - 19k - Cached - Similar pages

### Caching in ASP.NET: ASPAlliance

The third type of dependency is the key dependency. With it, a cache entry ... The Add/Insert method of the Cache class is used to add/insert an item into ...  
[aspalliance.com/795](http://aspalliance.com/795) - 56k - Cached - Similar pages

### 15 Seconds : Caching Oracle Data for ASP.NET Applications

Caching namespace. The Cache class provides methods for inserting data and retrieving data from the cache. The CacheDependency class enables dependencies to ...  
[www.15seconds.com/issue/031229.htm](http://www.15seconds.com/issue/031229.htm) - 84k - Cached - Similar pages

### Visual Web Developer 2005 Express Edition Beta Guided Tour

Moreover, the **SqlCommand** class inherently supports **dependency caching**. Simply instantiate the **SqlCacheDependency** class with a **SqlCommand** object and then use ...  
[www.asp.net/guidedtour2/s24.aspx](http://www.asp.net/guidedtour2/s24.aspx) - 15k - [Cached](#) - [Similar pages](#)

Google ►  
Result Page: 1 2 3 4 5 6 7 8 9 10 [Next](#)

New! Crack the Code: [Play the Da Vinci Code Quest on Google](#).

[Search within results](#) | [Language Tools](#) | [Search Tips](#) | [Dissatisfied? Help us improve](#)

[Google Home](#) - [Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2006 Google

**PORTAL**  
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search:  The ACM Digital Library  The Guide

cache store class dependency

## THE ACM DIGITAL LIBRARY

 [Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used cache store class dependency

Found 40,974 of 175,083

Sort results by

relevance 

[Save results to a Binder](#)

Try an [Advanced Search](#)

Display results

expanded form 

[Search Tips](#)

Try this search in [The ACM Guide](#)

Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale 

**1 Comparing the memory system performance of the HP V-class and SGI Origin 2000** 

 [multiprocessors using microbenchmarks and scientific applications](#)

Ravi Iyer, Nancy M. Amato, Lawrence Rauchwerger, Laxmi Bhuyan  
May 1999 **Proceedings of the 13th international conference on Supercomputing**

**Publisher:** ACM Press

Full text available:  [pdf\(1.13 MB\)](#) Additional Information: [full citation](#), [references](#), [index terms](#)



**2 Template instantiation for C++** 

 Glen McCluskey, Robert B. Murray

December 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 12

**Publisher:** ACM Press

Full text available:  [pdf\(693.62 KB\)](#) Additional Information: [full citation](#), [citations](#), [index terms](#)



**3 Limitation of superscalar microprocessor performance** 

 Thang Tran, Chuan-lin Wu

December 1992 **ACM SIGMICRO Newsletter**, **Proceedings of the 25th annual international symposium on Microarchitecture MICROS 25**, Volume 23 Issue 1-2

**Publisher:** IEEE Computer Society Press, ACM Press

Full text available:  [pdf\(495.55 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)



**4 Cache: Inter-reference gap distribution replacement: an improved replacement algorithm for set-associative caches** 

Masamichi Takagi, Kei Hiraki

June 2004 **Proceedings of the 18th annual international conference on Supercomputing**

**Publisher:** ACM Press

Full text available:  [pdf\(427.14 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

We propose a novel replacement algorithm, called Inter-Reference Gap Distribution Replacement (IGDR), for set-associative secondary caches of processors. IGDR attaches a weight to each memory-block, and on a replacement request it selects the memory-block

with the smallest weight for eviction. The time difference between successive references of a memory-block is called its Inter-Reference Gap (IRG). IGDR estimates the ideal weight of a memory-block by using the reciprocal of its IRG. To estimate ...

**Keywords:** cache memory, replacement algorithm, set-associative cache

**5 Classifying load and store instructions for memory renaming** 

Glenn Reinman, Brad Calder, Dean Tullsen, Gary Tyson, Todd Austin

May 1999 **Proceedings of the 13th international conference on Supercomputing**

**Publisher:** ACM Press

Full text available:  pdf(1.37 MB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**6 Fingerprinting: bounding soft-error detection latency and bandwidth** 

Jared C. Smolens, Brian T. Gold, Jangwoo Kim, Babak Falsafi, James C. Hoe, Andreas G. Nowatzky

October 2004 **ACM SIGPLAN Notices , ACM SIGARCH Computer Architecture News , ACM SIGOPS Operating Systems Review , Proceedings of the 11th international conference on Architectural support for programming languages and operating systems ASPLOS-XI**, Volume 39 , 32 , 38 Issue 11 , 5 , 5

**Publisher:** ACM Press

Full text available:  pdf(229.65 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Recent studies have suggested that the soft-error rate in microprocessor logic will become a reliability concern by 2010. This paper proposes an efficient error detection technique, called *fingerprinting*, that detects differences in execution across a dual modular redundant (DMR) processor pair. Fingerprinting summarizes a processor's execution history in a hash-based signature; differences between two mirrored processors are exposed by comparing their fingerprints. Fingerprinting tightly ...

**Keywords:** backwards error recovery (BER), dual modular redundancy (DMR), error detection, soft errors

**7 The effect of real data cache behavior on the performance of a microarchitecture that supports dynamic scheduling** 

Michael Butler, Yale Patt

September 1991 **Proceedings of the 24th annual international symposium on Microarchitecture**

**Publisher:** ACM Press

Full text available:  pdf(691.87 KB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**8 A comparison of two pipeline organizations** 

Michael Golden, Trevor Mudge

November 1994 **Proceedings of the 27th annual international symposium on Microarchitecture**

**Publisher:** ACM Press

Full text available:  pdf(849.79 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We examine two pipeline structures which are employed in commercial microprocessors. The first is the load-use interlock (LUI) pipeline, which employs an interlock to ensure correct operation during load-use hazards. The second is the address-generation interlock

(AGI) pipeline. It eliminates the load-use hazard, but has an address-generation hazard which requires an address-generation interlock for correct operation. We compare the performance of these two pipelines on existing binaries an ...

**Keywords:** RISC, cache memory, interlocks, memory system, pipelines

**9 Intelligent database caching through the use of page-answers and page-traces**

 Nabil Kamel, Roger King

December 1992 **ACM Transactions on Database Systems (TODS)**, Volume 17 Issue 4

**Publisher:** ACM Press

Full text available:  pdf(3.08 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



In this paper a new method to improve the utilization of main memory systems is presented. The new method is based on pre-storing in main memory a number of query answers, each evaluated out of a single memory page. To this end, the ideas of page-answers and page-traces are formally described and their properties analyzed. The query model used here allows for selection, projection, join, recursive queries as well as arbitrary combinations. We also show how to apply the approach under update ...

**Keywords:** artificial intelligence, databases, page access

**10 Effects of memory latencies on non-blocking processor/cache architectures**

 Koray Öner, Michel Dubois

August 1993 **Proceedings of the 7th international conference on Supercomputing**

**Publisher:** ACM Press

Full text available:  pdf(993.33 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)



In this paper, we introduce a simple hardware mechanism supporting non-blocking loads in conjunction with lockup-free caches to hide memory latencies in high-performance processors. The cache and processor cooperate on load misses so that the overall complexity of the non-blocking mechanisms in the cache and in the processor is greatly reduced. We use detailed simulations to evaluate the effectiveness of the architecture and of a simple compiler transformation at hiding miss latencies of up ...

**11 Conditional attribute grammars**

 John Tang Boyland

January 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 1

**Publisher:** ACM Press

Full text available:  pdf(222.04 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



Attribute grammars are a useful formalism for the specification of computations on structured terms. The classical definition of attribute grammars, however, has no way of treating conditionals nonstrictly. Consequently, the natural way of expressing many otherwise well-behaved computations involves a circularity. This article presents conditional attribute grammars, and extension of attribute grammars that enables more precise analysis of conditionals. In conditional attri ...

**Keywords:** attribute grammars, conditionals, demand evaluation, functional dependencies, language processor generators, nonstrict evaluation, static analysis

**12 Special issue on persistent object systems: Tigukat: a uniform behavioral objectbase**



**management system**

M. Tamer Özsu, Randal Peters, Duane Szafron, Boman Irani, Anna Lipka, Adriana Muñoz  
 July 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**,  
 Volume 4 Issue 3

**Publisher:** Springer-Verlag New York, Inc.

Full text available:  [pdf\(2.78 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

We describe the TIGUKAT objectbase management system, which is under development at the Laboratory for Database Systems Research at the University of Alberta. TIGUKAT has a novel object model, whose identifying characteristics include a purely behavioral semantics and a uniform approach to objects. Everything in the system, including types, classes, collections, behaviors, and functions, as well as meta-information, is a first-class object with well-defined behavior. In this way, the model abstr ...

**Keywords:** database management, objectbase management, persistent storage system, reflective system

**13 Service infrastructure and network management: Using code collection to support large applications on mobile devices** 

Lucian Popa, Irina Athanasiu, Costin Raiciu, Raju Pandey, Radu Teodorescu  
 September 2004 **Proceedings of the 10th annual international conference on Mobile computing and networking**

**Publisher:** ACM Press

Full text available:  [pdf\(252.95 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The progress of mobile device technology unfolds a new spectrum of applications that challenges conventional infrastructure models. Most of these devices are perceived by their users as "appliances" rather than computers and accordingly the application management should be done transparently by the underlying system unlike classic applications managed explicitly by the user. Memory management on such devices should consider new types of mobile applications involving code mobility such as mobile ...

**Keywords:** caching, code collection, garbage collection

**14 Caching function calls using precise dependencies** 

Allan Heydon, Roy Levin, Yuan Yu  
 May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35  
 Issue 5

**Publisher:** ACM Press

Full text available:  [pdf\(243.57 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper describes the implementation of a purely functional programming language for building software systems. In this language, external tools like compilers and linkers are invoked by function calls. Because some function calls are extremely expensive, it is obviously important to reuse the results of previous function calls whenever possible. Caching a function call requires the language interpreter to record all values on which the function call depends. For optimal caching, it is i ...

**15 Sensitivity analysis of a superscalar processor model** 

Y. Zhu, W. F. Wong  
 January 2002 **Australian Computer Science Communications , Proceedings of the seventh Asia-Pacific conference on Computer systems architecture - Volume 6 CRPITS '02**, Volume 24 Issue 3

**Publisher:** Australian Computer Society, Inc. , IEEE Computer Society Press

Full text available:  pdf(926.31 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Superscalar processors obtain their performance by exploiting instruction level parallelism in programs. Their performance is therefore limited by characteristics of programs and the design of the processor. Due to the complexity involved, estimating the performance of any superscalar processor design is a difficult task. Quick prediction of performance improvement arising from architecture modifications is even more difficult. In this paper, a model of superscalar processors using a network of ...

**Keywords:** queuing theory, superscalar processors

**16 Available instruction-level parallelism for superscalar and superpipelined machines** 

 N. P. Jouppi, D. W. Wall  
April 1989 **ACM SIGARCH Computer Architecture News, Proceedings of the third international conference on Architectural support for programming languages and operating systems ASPLOS-III**, Volume 17 Issue 2

Publisher: ACM Press

Full text available:  pdf(1.38 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Superscalar machines can issue several instructions per cycle. Superpipelined machines can issue only one instruction per cycle, but they have cycle times shorter than the latency of any functional unit. In this paper these two techniques are shown to be roughly equivalent ways of exploiting instruction-level parallelism. A parameterizable code reorganization and simulation system was developed and used to measure instruction-level parallelism for a series of benchmarks. Results of these si ...

**17 Static caching for incremental computation** 

 Yanhong A. Liu, Scott D. Stoller, Tim Teitelbaum  
May 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 20 Issue 3

Publisher: ACM Press

Full text available:  pdf(508.80 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

A systematic approach is given for deriving incremental programs that exploit caching. The cache-and-prune method presented in the article consists of three stages: (I) the original program is extended to cache the results of all its intermediate subcomputations as well as the final result, (II) the extended program is incrementalized so that computation on a new input can use all intermediate results on an old input, and (III) unused results cached by the extended progra ...

**Keywords:** caching, dependence analysis, incremental computation, incremental programs, intermediate results, memoization, optimization, program efficiency improvement, program transformation, static analysis

**18 Hierarchical materialisation of methods in object-oriented views: design, maintenance, and experimental evaluation** 

 Bartosz Bębel, Robert Wrembel  
November 2001 **Proceedings of the 4th ACM international workshop on Data warehousing and OLAP**

Publisher: ACM Press

Full text available:  pdf(3.02 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The application of materialised object-oriented views in object-relational data warehousing systems is promising. In this paper we propose a novel technique for the materialisation

of method results in object-oriented views, called *hierarchical materialisation*. When an object used to materialise the result of method  $m$  is updated, then  $m$  has to be recomputed. This recomputation can use unaffected intermediate materialised results of methods called from  $m$ , thus reducing ...

**Keywords:** method materialisation, object-oriented view, object-relational data warehouse, view materialisation

**19** Using dataflow analysis techniques to reduce ownership overhead in cache coherence protocols 

Jonas Skeppstedt, Per Stenström

November 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 6

Publisher: ACM Press

Full text available:  pdf(284.68 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#), [review](#)

In this article, we explore the potential of classical dataflow analysis techniques in removing overhead in write-invalidate cache coherence protocols for shared-memory multiprocessors. We construct the compiler algorithms with varying degree of sophistication that detect loads followed by stores to the same address. Such loads are marked and constitute a hint to the cache to obtain an exclusive copy of the block so that the subsequent store does not introduce access penalties. The simplest ...

**Keywords:** cache coherence, dataflow analysis, performance evaluation

**20** Verification techniques for cache coherence protocols 

Fong Pong, Michel Dubois

March 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 1

Publisher: ACM Press

Full text available:  pdf(1.25 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In this article we present a comprehensive survey of various approaches for the verification of cache coherence protocols based on state enumeration, (symbolic model checking, and symbolic state models. Since these techniques search the state space of the protocol exhaustively, the amount of memory required to manipulate that state information and the verification time grow very fast with the number of processors and the complexity of the protocol mechanism ...

**Keywords:** cache coherence, finite state machine, protocol verification, shared-memory multiprocessors, state representation and expansion

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2006 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)